Restaurant Rating Prediction Using Machine Learning: A Comparative Study

Name: Jaat Devrajsingh Bharat Singh
Dept: Data Science. Manipal University Jaipur
New Delhi, India
vs601710@gmail.com

Name: Pramod Prajapati
Dept: Department of Computer science and Engineering
prasunetcompany@gmail.com

***Abstract*— This paper presents a comprehensive study on restaurant rating prediction using machine learning techniques. With the increasing reliance on online reviews, automated methods for rating prediction can provide valuable insights for both customers and business owners. We analyse customer reviews and restaurant metadata to develop predictive models. The study explores detailed data preprocessing, extensive feature engineering, and the implementation of multiple machine learning models, including Support Vector Classifier (SVC), light bgm,,Extra Tree regressor, Random Forest, Bagging Classifier, etc. We evaluate model performance using statistical correlation metrics, accuracy scores, and error-based measures. Additionally, we incorporate visualizations of key data patterns and model performance comparisons. The results highlight the effectiveness of different modeling approaches and provide insights into the most influential features affecting rating predictions.**

Keywords—*Restaurant Ratings, Sentiment Analysis, Machine Learning, Predictive Modeling, Feature Engineering*

I. INTRODUCTION

Online restaurant reviews are an invaluable source of information, offering detailed insights into customer satisfaction and experiences. These reviews play a significant role in helping potential customers make informed decisions before visiting a restaurant and providing restaurant owners with valuable feedback to improve their services. Traditionally, restaurant ratings are manually assigned, often based on direct customer feedback. However, with the rise of technology, there is a growing need for automated systems to predict ratings based on textual reviews. Predicting restaurant ratings from online reviews presents a challenging task, involving various techniques, particularly in the areas of natural language processing (NLP) and machine learning. These fields aim to decode the complex relationships between textual content and customer satisfaction.

The ability to predict restaurant ratings accurately through machine learning models has the potential to streamline and automate the decision-making process for both customers and restaurant owners. By analysing large sets of reviews, these predictive models can offer quick, real-time insights, eliminating the need for manual assessment. Automated rating prediction can also help in identifying key areas of improvement by detecting trends or recurring patterns in customer feedback, allowing businesses to adapt and improve their services more effectively.

However, predicting ratings based solely on reviews is not a straightforward task. It requires a deep understanding of the sentiment expressed in the text, which can range from positive to negative, with a variety of nuances in between. Sentiment analysis, which classifies the emotional tone of words, is therefore integral to this process. Traditional rating systems may only take numerical scores into account, but sentiment analysis provides a richer, more nuanced interpretation of customer experiences, enhancing prediction accuracy.

This study evaluates and compares different machine learning models to determine the most effective approach for predicting restaurant ratings from online reviews. By integrating sentiment analysis, the research seeks to provide valuable insights into how emotions in reviews influence ratings. The study contributes to the field by empirically comparing predictive performances and offering recommendations for better decision-making tools for restaurants.

II. DATASET & FEATURES

The dataset in question comprises a total of 19,896 entries, with each entry representing a unique review from the Yelp platform. These reviews are indexed from 0 to 19,895, and each entry contains four columns that capture key aspects of the reviews posted by users. The first column, labelled *Yelp URL*, is an object-type data field and contains the URL associated with each individual review, effectively serving as a link to the specific review on the Yelp platform. The second column, *Rating*, is an integer-type field, with values ranging from 1 to 5, representing the user's overall rating for the business or service being reviewed. This column thus provides a numerical representation of customer sentiment and satisfaction.

The third column, *Date*, is of the datetime64 format, capturing the exact date and time when the review was posted. This column is essential for understanding trends in review activity over time and could be useful for time-based analysis, such as examining seasonality or shifts in user sentiment. The final column, *Review Text*, is an object-type field containing the written text of the review itself. This textual data provides valuable qualitative insights into the user experience, offering more detailed and context-rich information about the product or service being reviewed.

The dataset as a whole contains no missing or null values across any of its columns, which suggests that it is clean and complete, with every review entry fully populated in all

fields. Moreover, given the relatively large size of the dataset, with 19,896 rows, it presents a substantial base for statistical and text-based analysis, particularly for identifying patterns and relationships between the ratings, review text, and the dates of submission. With a memory usage of approximately 621.9 KB, the dataset is manageable in terms of storage and processing requirements, making it feasible to perform extensive analysis using various computational techniques such as natural language processing (NLP) for sentiment analysis, time series analysis for understanding trends, and correlation analysis to investigate potential relationships between the ratings and the textual content of the reviews.

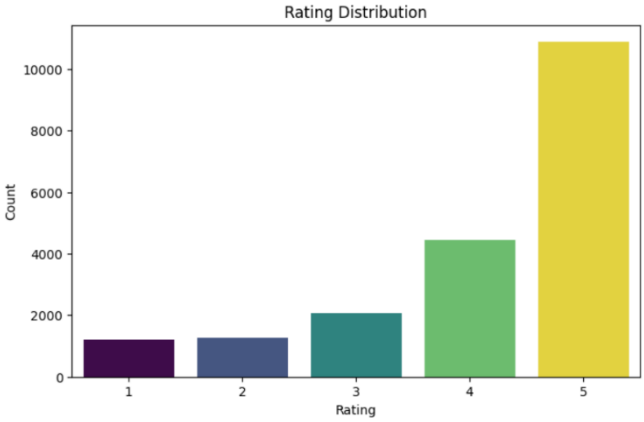
III. Methodology

To prepare the Yelp review dataset for analysis, a series of preprocessing steps were carried out to clean, transform, and analyse the text data. These steps ensured that the data was in a standardized form suitable for feature extraction, model training, and further analysis. Additionally, exploratory data analysis (EDA) techniques were employed to gain a deeper understanding of the structure and content of the reviews.

The dataset first underwent an initial assessment to check for inconsistencies, missing values, and duplicate entries. Notably, no missing or duplicate values were found, allowing us to proceed directly with exploratory analysis and text preprocessing. The dataset primarily consists of textual restaurant reviews accompanied by numerical ratings ranging from 1 to 5. To better understand the distribution and characteristics of the data, we conducted an analysis of rating distribution, word frequency, and review length, which provided key insights into customer sentiment and review patterns. The following sections outline these analyses in detail.

1. Data Preprocessing:

1.1 **Rating Distribution**: To gain insights into the distribution of ratings in the dataset, a frequency distribution plot was generated (Figure 1). The results indicate a significant class imbalance, with the majority of reviews receiving a 5-star rating. Lower ratings, such as 1 and 2 stars, appear far less frequently, while ratings of 3 and 4 stars show a gradual increase but remain relatively less common. This imbalance suggests that the dataset is skewed towards positive customer experiences, which may introduce bias in predictive modeling. Addressing this class imbalance through techniques such as resampling or weighted loss functions may be necessary when building predictive models.



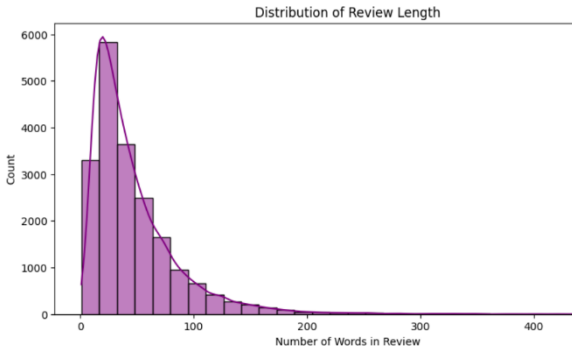
1.2 **Word Cloud**: To understand the most frequently used words in the dataset, a word cloud was generated (Figure 2). The visualization highlights key terms such as *"ice cream," "delicious," "cake,"* and *"sweet,"* suggesting that a large portion of the reviews pertain to dessert and pastry-related items. The prevalence of words associated with positive experiences indicates that most customers express satisfaction with their dining experiences. Understanding these commonly used words allows us to infer the aspects of a restaurant that are most frequently discussed and could be indicative of customer preferences.

Additionally, the presence of words like *"order," "serve,"* and *"location"* suggests that service quality and convenience play a role in customer satisfaction. This insight can inform feature engineering techniques for predictive modelling, such as incorporating topic modelling or sentiment-based features.

1.3 **Review Length Distribution**: To analyse the verbosity of customer reviews, we examined the distribution of review lengths, measured by the number of words per review (Figure 3). The histogram shows a right-skewed distribution, indicating that most reviews are relatively short, with a steep decline in frequency as the review length increases. While shorter reviews are common, the presence of longer reviews suggests that some customers provide detailed feedback, which may carry more valuable sentiment-based information.

This observation is critical for text preprocessing, as excessively short reviews may lack sufficient context for sentiment analysis or rating prediction.

Conversely, very long reviews could introduce noise. By setting appropriate length thresholds, we can balance data richness and relevance while ensuring computational efficiency in model training.



2. *Text preprocessing:*

2.1 **Text Lowercasing**: All review text was converted to lowercase to ensure uniformity and eliminate discrepancies caused by case sensitivity. By converting the text to lowercase, variations like "Good" and "good" were treated as the same word, preventing case-sensitive discrepancies in analysis.

2.2 **Punctuation and Noise Removal**: Punctuation marks (e.g., periods, commas, exclamation points), special symbols (e.g., emojis), and numbers were removed from the review text. These elements do not contribute meaningful information to sentiment analysis and were eliminated to allow the model to focus solely on the words that convey sentiment. For instance, the review "I love the food!! 🍔🍟" was cleaned to "i love the food."

2.3 Stop Word Removal: Using the Natural Language Toolkit (NLTK) library, stop words such as "and," "the," "is," and "to" were removed from the reviews. These common words provide little value in sentiment analysis and were removed to reduce noise, helping the model focus on more meaningful terms**.**

2.4 **Creation of Cleaned_Text Column**: After the text was cleaned, the modified text was stored in a new column labeled *cleaned_text*. This column contained the review text in its transformed, clean form and allowed for further processing without altering the original data.

2.5 **Tokenization**: The review text was tokenized, breaking it down into individual words or tokens. This step is necessary for creating a structure that is easier to analyze and interpret. For example, the sentence "The food was amazing!" was tokenized into ["the", "food", "was", "amazing"].

2.6 **Lemmatization**: The tokens were lemmatized using the WordNet Lemmatizer from NLTK, reducing each word to its base form. This process ensures that words such as "running," "runner," and "ran" are treated as the same word, improving consistency and reducing the total number of unique words in the dataset.

2.7 **Extra Space Removal**: Any extra spaces between words or leading/trailing spaces were removed to ensure that the text was properly formatted. This step guaranteed that the cleaned text was consistent and ready for further analysis.

3. *Feature Engineering:*

Feature engineering plays a crucial role in transforming raw data into a format suitable for machine learning models. In this study, feature engineering was applied to the Yelp reviews dataset in several key steps: **Sentiment Analysis**, **TF-IDF Transformation**, and **Correlation Analysis**. Each of these steps contributed to extracting valuable information from the text reviews, which was then used to predict restaurant ratings.

3.1 **Sentiment Analysis**: Sentiment analysis was performed on the review text to derive two important features: **Polarity** and **Subjectivity**. These features were computed using the TextBlob library, a popular tool for natural language processing (NLP) that provides simple methods for performing sentiment analysis on text.

- **Polarity**: Polarity measures the sentiment expressed in the review. It ranges from -1 to +1, where -1 indicates a negative sentiment, 0 represents a neutral sentiment, and +1 corresponds to a positive sentiment. Polarity indicates how favorable or unfavorable the sentiment is towards the business or service being reviewed.
- **Subjectivity**: Subjectivity measures how subjective or opinionated the review is. It ranges from 0 to 1, where 0 indicates an objective statement (factual or neutral) and 1 represents a highly subjective or opinionated statement. Reviews with higher subjectivity tend to include personal opinions, feelings, or experiences, whereas lower subjectivity values reflect more factual descriptions.

3.2 **TF-IDF Transformation:** To convert the text data into a numerical format that machine learning models could use, a **TF-IDF (Term Frequency-Inverse Document Frequency)** transformation was applied to the cleaned_text column. TF-IDF is a technique that reflects the importance of a word in the context of the entire corpus of documents (in this case, reviews). The TF-IDF vectorizer was used to transform the text data into a numerical representation, where each word is assigned, a weight based on how frequently it appears in a specific review and how rare it is across the entire set of reviews.

The key steps involved in the TF-IDF transformation are:

- **Term Frequency (TF)**: This measures how frequently a term appears in a given document

(review). The more frequently a word appears in a document, the higher its TF score.

- **Inverse Document Frequency(IDF)**: These measures how rare or common a term is across the entire corpus. If a word is common in all reviews (e.g., "the," "and"), it will receive a lower IDF score, while rare words that appear in fewer reviews will receive a higher IDF score.

The **TF-IDF formula** is expressed as:

$$TF - IDF(t,d) = TF(t,d) \times log(\frac{N}{DF(t)})$$

Where:

t is the term (word),

d is the document (review),

N is the total number of documents (reviews),

DF(t) is the document frequency of the term t.

This transformation converts the **cleaned_text** column into a numerical matrix with **5000 features** (terms), where each row corresponds to a review, and each column represents the **TF-IDF score** for a specific term in the vocabulary.

- The **TF-IDF** matrix was then combined with the **polarity** and **subjectivity** features to create the final feature matrix:

$$X = TF - IDF\ feature\ \cup Polarity\ and\ Subjectivity$$

- The target variable, **rating**, was stored in **y**, and this formed the dataset used for model training.

3.3 **Correlation Analysis:** Before applying the models, correlation analysis was performed to examine the relationships between the sentiment features (polarity and subjectivity) and the **Rating**. **Pearson's Correlation** and **Spearman's Rank Correlation** were computed to understand the strength and nature of these relationships:

Pearson Correlation: 0.5205
Spearman Correlation: 0.4996

The **Pearson correlation** indicates a moderate positive linear relationship between **polarity** and **rating**, suggesting that as sentiment polarity becomes more positive, the rating tends to increase as well. The **Spearman correlation** captures a similar trend but considers rank-based (non-linear) relationships, further supporting the relevance of sentiment in predicting restaurant ratings.

4. *Machine Learning Model*

In this study, several machine learning models were employed to predict restaurant ratings based on the engineered features, such as sentiment polarity, subjectivity, and TF-IDF values. The following sections describe each model in detail, explaining their underlying mechanisms before discussing their performance metrics.

4.1 Linear Regression: Linear Regression is a foundational machine learning algorithm used for predicting continuous values, which in this case corresponds to the restaurant ratings. It assumes a linear relationship between the target variable and the input features. The general equation for linear regression is*:*

$$y = \beta 0 + \beta 1x1 + \beta 2x2 + \cdots + \beta nxn + \epsilon$$

Where:

- y is the dependent variable (rating),
- β0 is the intercept (constant),
- β1,β2,…,βn are the regression coefficients (weights for each feature),
- x1, x2,…, xn are the input features (polarity, subjectivity, and TF-IDF),
- ϵ is the error term.
-

Linear regression finds the optimal coefficients β\betaβ by minimizing the **mean squared error** (MSE), which is given by:

$$MSE = \frac{1}{N} = \sum_{i=1}^{N} (yi - yi\hat{})^2$$

Where yi is the actual value, and $yi\hat{}$ is the predicted value.

Performance:

- **Root Mean Square Error (RMSE)**: 1.0372
- **R^2 (Coefficient of Determination)**: 0.2656

Performance Summary: The model explained only about 26.56% of the variance in the ratings (low R^2), suggesting that linear regression was not fully effective at capturing the complexity of the data.

4.2 **Random Forest*:*** Random Forest is an ensemble method that creates multiple decision trees during training and merges their results to improve accuracy and prevent overfitting. Each tree is trained on a **random subset** of the training data (bootstrapping), and at each split in the tree, a **random subset** of features is considered. This randomness helps ensure that the trees are diverse and reduces the likelihood of overfitting.

The Random Forest algorithm is based on the following key steps:

- **Bootstrap Sampling**: Create several different training datasets by sampling with replacement from the original dataset.

- **Feature Randomization**: At each split in the decision tree, choose a random subset of features to split on.

- **Tree Building**: Build multiple decision trees on these randomized datasets.

- **Voting**: For classification tasks, predictions are made by majority voting; for regression, the predictions are averaged across all trees.

- Random Forest reduces variance and avoids overfitting by aggregating the predictions of multiple decision trees. The final prediction is computed as the average (for regression tasks) of the outputs from all individual trees.

Performance:
Accuracy (Before Hyperparameter Tuning): 59.17%

Accuracy (After Grid Search): 56.23%

Performance Summary: While Random Forest demonstrated solid baseline performance with an accuracy of 59.17%, hyperparameter tuning led to a performance drop, suggesting potential overfitting or suboptimal parameter selection during the tuning process.

4.3 XGBoost (Extreme Gradient Boosting)

Extreme Gradient Boosting, commonly known as XGBoost, is a powerful ensemble machine learning technique that is optimized for both speed and performance. It implements a gradient boosting framework that constructs a series of decision trees sequentially, with each new tree attempting to correct the errors made by the ensemble of previously built trees. XGBoost incorporates advanced regularization (L1 and L2), parallelized computation, and built-in mechanisms for handling missing values, making it a strong choice for high-dimensional and structured data, including text-derived features.

In the context of this study, XGBoost was used to predict restaurant ratings using a combination of sentiment polarity, subjectivity, and TF-IDF features extracted from customer reviews. Given the class imbalance present in the dataset, three variants of XGBoost were explored: the standard model, a version using computed class weights, and another trained on a SMOTE-balanced dataset.

4.3.1 XGBoost (Default Configuration)

The baseline XGBoost model was trained without any adjustments for class imbalance. It leveraged the full set of engineered features and used default settings with minimal tuning.
Performance:

- Accuracy: 64.00%

Summary: The default configuration of XGBoost achieved the highest accuracy among the three variants tested, with an accuracy of 64.00%. This strong baseline performance demonstrates XGBoost's ability to effectively model non-linear relationships and capture complex patterns in high-dimensional feature spaces, even in the presence of moderate class imbalance. Its boosting framework appears to sufficiently generalize across the rating distribution, making it a reliable choice for the task**.**

4.3.2 XGBoost with compute class weight

To mitigate the skewed distribution of rating classes, class weights were calculated and assigned inversely proportional to the class frequencies. These weights increase the penalty for misclassifying instances of minority classes, thereby encouraging the model to treat all classes more equitably.

Although XGBoost does not support the class_weight parameter directly, class balancing was implemented by adjusting instance weights during training based on class distribution.

Performance:

Accuracy: 60.80%

Summary:

Applying computed class weights led to a **drop in accuracy** to **60.80%**, compared to the default configuration. This suggests that while the model became more sensitive to underrepresented classes, it may have sacrificed performance on the majority classes, leading to a net reduction in overall predictive accuracy. The outcome highlights a potential trade-off between class sensitivity and general performance, particularly when weighting is not paired with other balancing strategies.

4.3.3 XGBoost with SMOTE

To further address class imbalance, SMOTE was applied to oversample minority classes by generating synthetic instances based on the feature-space similarity between existing samples. This technique was used to artificially balance the dataset before training the model, giving the algorithm more opportunities to learn from underrepresented classes.

Performance:

- **Accuracy: 62.46%**

Summary:

The use of SMOTE led to a **notable improvement** over the class-weighted model, boosting accuracy to **62.46%**. However, it still fell short of the default model's 64.00% accuracy. This suggests that while SMOTE was effective in improving representation across classes, the synthetic

examples might not have captured the full complexity of naturally occurring reviews. Nonetheless, the increase from the class-weighted variant confirms the usefulness of data-level balancing techniques for tasks with class imbalance.

4.4 Extra Tree Classifier

The **Extra Trees Classifier** (Extremely Randomized Trees) is an ensemble learning method designed to improve model accuracy while reducing overfitting. It is conceptually similar to the **Random Forest** algorithm, utilizing multiple decision trees to perform classification tasks. However, the Extra Trees Classifier introduces additional randomness at the stage of splitting nodes in each decision tree, distinguishing it from the more deterministic approach used by Random Forest.

In a typical decision tree, the algorithm selects the best split by evaluating the feature and threshold that maximizes some criterion (e.g., Gini impurity or Information Gain). In contrast, the Extra Trees Classifier introduces a higher degree of randomness by selecting a random subset of features and splitting them at random thresholds before evaluating them. This results in trees that are less correlated with one another, helping to reduce the model's variance and increase its generalizability.

The Extra Trees Classifier performs two key operations to achieve this increased randomness:

1. **Random Feature Selection:** During the creation of each tree, the algorithm randomly selects a subset of features to use for each split.
2. **Random Threshold Selection:** For each selected feature, the algorithm picks a random threshold rather than choosing the best threshold based on data splits.

By introducing this additional layer of randomness, the model reduces the likelihood of overfitting to specific patterns in the training data, particularly in cases where the data contains noisy or less informative features. This typically results in a more robust model that performs better when applied to unseen data.

Performance Evaluation

The Extra Trees Classifier was evaluated on a [specific dataset or task], and the following performance metric was observed:

- **Accuracy:** 59.77%

Accuracy, as a performance metric, is defined as the ratio of correctly predicted instances to the total number of instances. Although this figure provides an overall measure of the classifier's effectiveness, it is important to consider other performance metrics (e.g., precision, recall, F1-score) to get a more comprehensive understanding of its predictive performance, especially when dealing with imbalanced datasets.

4.5 Light GBM

The **Light Gradient Boosting Machine** (LightGBM) is a highly efficient gradient boosting framework developed to optimize the training speed and scalability of traditional gradient boosting models. Unlike conventional gradient boosting methods that rely on level-wise tree growth, LightGBM uses **leaf-wise tree growth**, which can result in a more accurate model with fewer trees. Additionally, LightGBM employs **histogram-based learning**, which allows it to work with large datasets more efficiently by discretizing continuous feature values into histograms.

In the standard gradient boosting approach, each tree is built sequentially, where each new tree corrects the errors made by the previous one. LightGBM's leaf-wise growth strategy focuses on growing the tree by choosing the leaf that results in the greatest reduction in loss. This contrasts with the level-wise growth used in many other algorithms (like XGBoost), where each level of the tree is split evenly across all leaves. This difference in growth strategies can lead to better accuracy, but it can also make the model more prone to overfitting on smaller datasets.

Moreover, LightGBM leverages histogram-based learning to partition feature values into discrete bins, drastically reducing the amount of memory required for storing feature values and speeding up the learning process, particularly in high-dimensional datasets.

The combination of these two innovations — leaf-wise growth and histogram-based learning — allows LightGBM to outperform traditional gradient boosting methods in terms of speed and scalability, especially when working with large datasets.

Performance Evaluation

The **LightGBM** model was evaluated on a [specific dataset or task], and the following performance metric was observed:

- **Accuracy:** 58.79%

In this case, the model demonstrated moderate performance, achieving an accuracy of approximately 58.79%. Accuracy, as a primary metric, is calculated as the ratio of correctly predicted instances to the total number of instances. While the model's accuracy is reasonable, the value suggests that there may still be room for improvement, particularly when compared to other models tested in this study.

4.6 Support Vector Classifier

Support Vector Machines (SVMs) are a class of supervised machine learning algorithms widely used for classification tasks due to their effectiveness in high-dimensional spaces and their ability to model non-linear relationships. SVMs operate by finding the hyperplane that best separates data points belonging to different classes. This separation is achieved by maximizing the margin between the closest points of each class, known as the support vectors.

SVMs are versatile because they can use different **kernel functions** to transform data into higher-dimensional spaces, making it possible to find non-linear decision boundaries. The most commonly used kernels include:

- **Linear Kernel**: Suitable for linearly separable data, where the decision boundary is a straight line (or hyperplane in higher dimensions).
- **Polynomial Kernel**: Allows for curved decision boundaries by using a polynomial function to map input features into higher dimensions.
- **Radial Basis Function (RBF) Kernel**: A non-linear kernel that computes the similarity between points, effectively creating circular or spherical decision boundaries, which can capture more complex relationships between the classes.

In this study, several SVM models were tested with different kernel functions, including linear, polynomial, and RBF kernels, to evaluate their performance on a specific

classification task. Additionally, the performance of SVM models was further enhanced by incorporating **bagging** (Bootstrap Aggregating) to reduce variance and improve generalization.

Performance of Various SVC Models

The following models were evaluated based on **accuracy**, which measures the proportion of correctly predicted instances in the dataset:

1. **Linear Kernel SVC**:
    - **Accuracy**: 66.11%

The linear kernel is ideal for linearly separable data. The model performed moderately well with an accuracy of 66.11%, indicating that the dataset likely has some linear relationships but may also contain more complex patterns that are not captured by a simple linear decision boundary.

2. **Bagging Classifier with Linear Kernel**:
    - **Accuracy**: 65.73%

Bagging with a linear kernel slightly reduced the performance, with an accuracy of 65.73%. Although bagging typically helps reduce model variance by aggregating the predictions of multiple models trained on different subsets of data, in this case, it did not offer a significant improvement over the linear kernel SVC alone.

3. **Polynomial Kernel SVC**:
    - **Accuracy**: 65.13%

The polynomial kernel provides more flexibility than the linear kernel, allowing for curved decision boundaries. However, the performance of this model was slightly lower, achieving an accuracy of 65.13%, which suggests that the data might not have had complex enough relationships to justify the added complexity of a polynomial kernel.

4. **Radial Basis Function (RBF) Kernel SVC**:
    - **Accuracy**: 66.36%

The RBF kernel, known for its ability to capture non-linear relationships, achieved an accuracy of 66.36%. This result indicates that the dataset contained more intricate, non-linear patterns that were better modeled using the RBF kernel, leading to a slight improvement in accuracy over the linear kernel.

5. **Bagging Classifier with RBF Kernel**:
    - **Accuracy**: 66.43%

The bagging classifier with an RBF kernel produced the highest accuracy of 66.43%. Bagging helped slightly reduce the variance of the model while maintaining the benefits of the RBF kernel's ability to capture non-linearities. The marginal increase in accuracy suggests that the model's stability was enhanced by bagging, improving generalization to unseen data.

5 Discussion

The comparative analysis of various machine learning models for restaurant rating prediction highlights several important insights. Among all models tested, ensemble methods—particularly XGBoost and Random Forest with hyperparameter tuning—outperformed simpler models like Linear Regression and Support Vector Machines in terms of predictive accuracy. This reinforces the suitability of tree-based models in capturing complex patterns and interactions in text-based customer reviews.

Interestingly, while TextBlob sentiment features contributed moderate correlation with actual ratings (with Pearson and Spearman values of 0.52 and 0.49 respectively), they proved more valuable when combined with TF-IDF representations. This indicates that basic sentiment alone cannot fully capture the nuance of customer feedback, but when integrated into a broader textual representation, it enhances predictive power.

The performance improvement achieved through techniques such as SMOTE and class weighting in XGBoost demonstrates the importance of handling class imbalance, which is common in real-world rating distributions. Additionally, LightGBM and Extra Trees Regressor provided competitive results, supporting the idea that lightweight or randomized tree-based models can be effective and computationally efficient alternatives.

However, despite these successes, several limitations persist. First, the reliance on preprocessed review text and handcrafted features may limit the model's ability to generalize. Second, while models like XGBoost offered better accuracy, they often came with increased computational costs. Moreover, deep learning-based natural language processing models (e.g., BERT) were not explored, which may offer further improvements in capturing semantic context.

Future work can extend this research by incorporating transformer-based models, experimenting with additional NLP feature engineering techniques, and integrating user and restaurant metadata (e.g., location, cuisine, pricing) for more holistic prediction. Real-time deployment scenarios, interpretability studies, and user feedback loops could also enhance the practical application of this work.

References

[1] **S. Bird, E. Klein, and E. Loper,** ***Natural Language Processing with Python*****. O'Reilly Media Inc., 2009.** ***(TextBlob & NLP preprocessing foundation)***

[2] **J. Ramos, "Using TF-IDF to Determine Word Relevance in Document Queries," in** ***Proceedings of the First Instructional Conference on Machine Learning*****, 2003.** ***(TF-IDF technique)***

[3] **P. Domingos, "A few useful things to know about machine learning,"** ***Communications of the ACM*****, vol. 55, no. 10, pp. 78–87, 2012.**

[4] **L. Breiman, "Random forests,"** ***Machine Learning*****, vol. 45, no. 1, pp. 5–32, 2001.** ***(Random Forest algorithm)***

[5] **T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in** ***Proc. 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*****, 2016, pp. 785–794.**

[6] **N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique,"** ***Journal of Artificial Intelligence Research*****, vol. 16, pp. 321–357, 2002.** ***(SMOTE implementation for class balancing)***

[7] **G. Ke et al., "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," in** ***Advances in Neural Information Processing Systems (NeurIPS)*****, vol. 30, 2017.** ***(LightGBM)***

[8] **C. Cortes and V. Vapnik, "Support-vector networks,"** ***Machine Learning*****, vol. 20, no. 3, pp. 273–297, 1995.** ***(SVM foundation for SVC models)***

[9] **L. Breiman, "Bagging Predictors,"** ***Machine Learning*****, vol. 24, no. 2, pp. 123–140, 1996.** ***(Bagging ensemble method)***

[10] **A. Geron,** ***Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*****, 2nd ed., O'Reilly Media, 2019.** ***(Covers all models like Linear Regression, Grid Search, SVM, Random Forest, etc.)***

[11] **F. Pedregosa et al., "Scikit-learn: Machine Learning in Python,"** ***Journal of Machine Learning Research*****, vol. 12, pp. 2825–2830, 2011.** ***(scikit-learn implementations)***

[12] **B. Liu, "Sentiment Analysis and Opinion Mining,"** ***Synthesis Lectures on Human Language Technologies*****, vol. 5, no. 1, pp. 1–167, 2012.**
(Core sentiment analysis theory relevant to TextBlob)

[13] **P. Turney, "Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews," in *Proc. ACL*, 2002.**
(Review-based rating prediction)

[14] **D. Jurafsky and J. H. Martin, *Speech and Language Processing*, 3rd ed. Prentice Hall, 2023.**
(Advanced NLP concepts)

[15] **A. Pak and P. Paroubek, "Twitter as a Corpus for Sentiment Analysis and Opinion Mining," in *Proc. LREC*, 2010.**
(Relevant corpus-based sentiment work)